# D2.4.7 Web Service Invocation and Interoperation (v1)

**Paavo Kotinurmi, Tomas Vitvar**
**(National University of Ireland, Galway)**

**with contributions from:**

**Mick Kerrigan, Jana Viskova (National University of Ireland, Galway)**

**Abstract.**
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB Deliverable D2.4.7 (WP2.4)

The goal of this deliverable is to provide guidelines for achieving interoperation and invocation of services in the inter-enterprise integration settings by means of the Semantic Web Services concepts and technologies, namely WSMO, WSML and WSMX. We address interoperation in technical, data and process levels. With emphasis on technical interoperation and within the context of a case scenario we propose a solution for integration of business partners using different B2B protocols by means of the SWS technologies.

Keyword list: Web Services, Invocation, Interoperation, Mediation, B2B integration, RosettaNet

| | |
|---|---|
| Document Identifier | KWEB/2004/D2.4.7/1.0 |
| Project | KWEB EU-IST-2004-507482 |
| Version | v1.0 |
| Date | February 13, 2006 |
| State | final |
| Distribution | public |

## Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

**University of Innsbruck (UIBK) - Coordinator**
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

**École Polytechnique Fédérale de Lausanne (EPFL)**
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

**France Telecom (FT)**
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

**Freie Universität Berlin (FU Berlin)**
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

**Free University of Bozen-Bolzano (FUB)**
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

**Institut National de Recherche en Informatique et en Automatique (INRIA)**
ZIRST - 655 avenue de l’Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

**Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)**
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

**Learning Lab Lower Saxony (L3S)**
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

**National University of Ireland Galway (NUIG)**
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Tomas Vitvar
E-mail address: tomas.vitvar@deri.ie

**The Open University (OU)**
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

**Universidad Politécnica de Madrid (UPM)**
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

**University of Karlsruhe (UKARL)**
Institut für Angewandte Informatik und Formale Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

**University of Liverpool (UniLiv)**
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

**University of Manchester (UoM)**
Room 2.32. Kilburn Building, Department of Computer Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

**University of Sheffield (USFD)**
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

**University of Trento (UniTn)**
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

**Vrije Universiteit Amsterdam (VUA)**
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

**Vrije Universiteit Brussel (VUB)**
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

# Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

École Polytechnique Fédérale de Lausanne
France Telecom
Freie Universität Berlin
National University of Ireland Galway
University of Innsbruck
University of Liverpool
University of Manchester
University of Trento

# Changes

| Version | Date | Author | Changes |
|---|---|---|---|
| 0.1 | 20.04.05 | Mick Kerrigan | Document Creation |
| 0.2 | 22.11.05 | Paavo Kotinurmi | First draft |
| 0.5 | 15.12.05 | Paavo Kotinurmi | First complete version |
| 0.9 | 7.1.06 | Tomas Vitvar | Amended structure and text |
| 1.0 | 13.2.06 | Paavo Kotinurmi | Addressed the review comments |

# Executive Summary

The attention to interoperation and invocation of web services is paid in a number of research projects. Interoperation and invocation of web services can be distinguished at the levels of (1) technical interoperation, (2) data (semantic) interoperation, and (3) process interoperation. Technical level interoperation means being able to handle different communication protocols and across services that are made using different languages. Data level interoperation must be ensured when different meanings (semantics) of messages are used for communication. Process level interoperation must be ensured when different communication patterns (choreographies) are used during communication.

With respect to all interoperation levels and within the context of ongoing work on interoperation and invocation in other research projects, we are focused in this deliverable on particular problem of technical interoperation taking into account different business to business protocols used by different parties involved in communication. These tasks are within the SWS execution environment covered mainly by adapters and techniques of ontologizing of particular e-business or communication standards. With this respect, this deliverable presents a scenario for B2B integration, where a buyer organization communicates with multiple suppliers, which all behave a bit differently. The suppliers support different standards for this communication of purchasing process information. In the scenario, the requests for quotes (RFQ) and purchase order (PO) processes are used. Currently, setting up a B2B integration takes a lot of time for companies, even six months to set up one connection. With the help of SWS technologies, this time can be significantly reduced. The interoperation aspects are handled at the buyer organization point of view, where the manufacturer's use of Semantic Web Service (SWS) technologies enables it to communicate with all the suppliers. In this deliverable it is shown, how SWS execution environment could be used to help this interoperation successful.

# Contents

# Chapter 1

# Introduction

## 1.1 Goal of the Deliverable

The goal of this deliverable is to provide **guidelines for achieving interoperation and invocation of services in the inter-enterprise integration settings** by means of the Semantic Web Services concepts and technologies, namely WSMO, WSML and WSMX. With this respect we address all interoperation levels (technical, data, and process) and with the focus on the technical interoperation we show how this interoperation can be achieved with use of SWS technology between different business partners using different B2B protocols and RosettaNet B2B protocol in particular.

This is the first version of the deliverable which contributes to the work done within the WP2.4 Semantic Web Services including requirements for semantic description of web services, conceptual and formal framework for the Semantic Web Services, guidelines for the integration of agent-based services and web-based services, theoretical integration of Web Service discovery and composition, and reputation mechanism. In general, the work in this deliverable contributes and complement the work done for the SWS specifications around Web Service Modelling Ontology (WSMO) [RLK04] and Web Service Modelling Language (WSML) [dBL+05], and in particular for the Web Service Modelling eXecution environment (WSMX) [ZMH05].

## 1.2 Overview of the Deliverable

Further in this chapter we present the terms used throughout the deliverable and we describe semantic web services concepts and technologies according to the WSMO and WSMX specifications. We also explain what we understand by interoperation and invocation of web services. In chapter 2 we describe a motivation use case scenario which we will use as a running example in this deliverable. In chapter 3 we describe guidelines for achieving interoperation of services and in detail demonstrate technical-level interoperation on integration of RosettaNet B2B protocol with WSMX. In chapter 4 we conclude the deliverable and describe our future work.

## 1.3 Terminology

In this section we present major terms used throughout the deliverable.

- **B2B protocol** - a standard for B2B communication such as RosettaNet or EDI, that define how B2B integration takes place.
- **EDI** - Electronic Data Interchange. Standard for B2B integration, i.e. a B2B protocol. First EDI standards are from 1970's and they have been used a lot in B2B integration.
- **ERP** - Enterprise Resource Planning. An information system used in enterprises for storing financial, logistics, material etc. information.
- **Idoc** - SAP intermediate documents. A format for connecting SAP ERP system to other systems.
- **Organization** - refers to the buyer in the scenario of this deliverable.
- **Partner** - Refers to the potential suppliers (sellers) in the scenario of this deliverable.
- **PIP** - Partner Interface Process. RosettaNet standard for B2B message exchange. Contain both process and message guidelines.
- **PO** - Purchase Order. A process used in the scenario to get the information for purchase the needed devices.
- **RFQ** - Request For Quote. A process used in the scenario prior to PO to get the price and shipment information.
- **RNIF** - RosettaNet Implementation Framework. RosettaNet standard for secure transport of RosettaNet PIPs. Guides how messages are paced, secured and acknowledged, when exchanged over the Internet from peer-to-peer.
- **SWS** - Semantic Web Service. Semantically enabled web services are represented so that computers can understand them.
- **UDDI** - Universal Description Discovery and Integration. Repository where WSDL documents are often stored.
- **WSDL** - Web Services Description Language. Way to describe the web service information.

- **WSML** - Web Service Modelling Language. WSML is a language that formalizes the WSMO and WSML is used in WSMX.
- **WSMO** - Web Service Modelling Ontology. WSMO adheres to the principles of loose coupling of services and strong mediation among them. WSMO defines an underlying model for WSMX.
- **WSMX** - Web Service Modelling eXecution environment. WSMX is the reference implementation of WSMO. It is an execution environment for business application integration where enhanced web services are integrated for various business applications.

## 1.4 Semantic Web Services

Web Services are small units of functionality, which are made available by service providers for use in larger applications. The intention when developing Web Services was to reduce the overhead needed to integrate functionality from multiple providers. Communication with Web Services is usually achieved using the SOAP protocol [GHM$^+$03]. SOAP is an XML based protocol for communication between distributed environments. Descriptions of the interfaces of the Web Services are described using the Web Service Description Language (WSDL) [CCMW01]. WSDL documents are generally stored in a Universal Description Discovery and Integration (UDDI) [1] repository where services can be discovered by end-users.

A major issue with Web Services is that their interfaces are only explained in a syntactic fashion using WSDL documents. While it is possible for computers to process these documents it is not possible for computers to 'understand' them. Inevitably a human is required to find WSDL documents in a UDDI repository, study these documents and understand them in order to integrate the Web Services into a system. While Web Services have indeed reduced the overhead needed to integrate functionality from multiple providers, extensive human interaction is still required in the process. Semantically-enabled web services are forming the research area known as Semantic Web Services. Semantic Web Services complement standards around WSDL, SOAP and UDDI with aim to enable total or partial automation of tasks such as discovery, selection, composition, mediation, invocation and monitoring of services. The research lies in definition and development of concepts, ontologies, languages and technologies for Semantic Web Services. A number of initiatives exist looking at how to create and manage semantic description for Web Services including OWL-S [Mar04], Meteor-S [POSV04], WSDL-S [WSD05] and WSMO [RLK04].

One of the major added values of the Semantic Web Services according to the WSMO concepts lies in the (semi) automated interoperation and invocation of services. By in-

[1] http://www.uddi.org/

teroperation and invocation of services we understand the ability of different services to communicate at different levels, namely *technical*, *data* and *process* levels.

1. Technical level – interoperation must be ensured when different protocols are used for communication as well as different languages. Interoperation at this level is achieved by adaptation of protocols and languages used, that is by their syntactical translations as well as grounding to underlying communication protocols (invocation of WS).
2. Data level – interoperation must be ensured when different meanings (semantics) of messages are used for communication. It is achieved by data mediation with use of ontology integration techniques, such as ontology mapping/aligning.
3. Process level –interoperation must be ensured when different communication patterns (choreographies) are used during communication. It is achieved by process mediators providing functionality for a runtime analysis of two given patterns, and compensates for the possible mismatches which may appear.

### 1.4.1 WSMO, WSML, and WSMX

Web Services Modeling Ontology (WSMO) has its conceptual basis in the Web Service Modeling Framework (WSMF) [FB02], which adheres to the principles of loose coupling of services and strong mediation among them. WSMO defines an underlying model for the WSMX Semantic Web Services execution environment [ZMH05] as well as draws up requirements for a WSML ontology language [dBL+05] used for formal description of WSMO elements. Thus, WSMO, WSML and WSMX form a complete framework to deal with all aspects of the Semantic Web Services.

WSMO top-level conceptual model is composed of *Ontologies*, *Goals*, *Web Services* and *Mediators*.

**Ontologies** provide formal explicit specification of shared conceptualization that is formal semantics of information used by other components (goals, web services, and mediators). WSMO specifies the following constituents as part of the description of an ontology: *concepts*, *relations*, *functions*, *axioms*, and *instances* of concepts and relations, as well as *non-functional properties*, *imported ontologies*, and *used mediators*. The latter allows the interconnection of different ontologies by using mediators that solve terminology mismatches.

**Goals** provide description of objectives of a service requester (user) that he or she wants to achieve. WSMO goals are described in terms of desired information as well as "state of the world" which must result from execution of a given service. In WSMO, a goal is characterized by a set of *non-functional properties, imported ontologies, used mediators*, a *requested capability* and a *requested interface* (these definitions are the same as for web services).

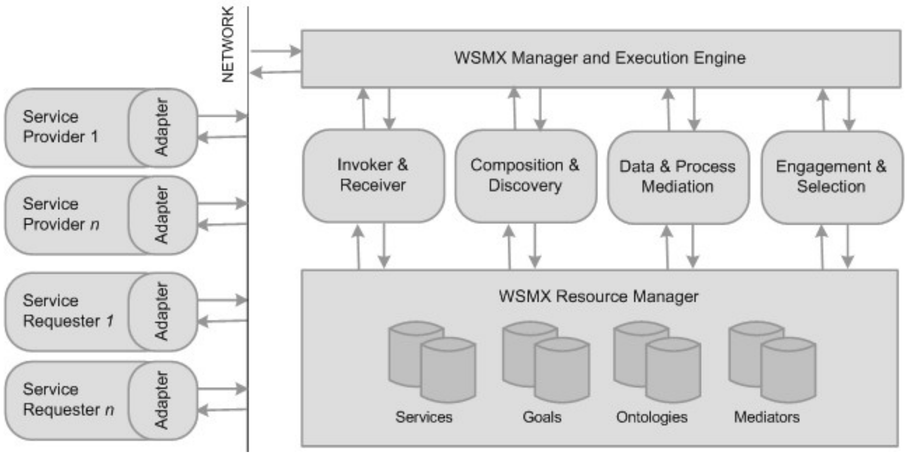

Figure 1.1: WSMX Architecture

**Web Services** provide a functionality for a certain purpose, which must be semantically described. Such description includes *non-functional properties*, *imported ontologies*, *used mediators*, *capability* and *interfaces*. *Capability* of a web service is modeled by *preconditions* and *assumptions* for the correct execution of the web service as well as *postconditions* and *effects* resulting from this execution. The interface for every web service is modeled as *choreography* describing communication pattern (interactions) with this web service and *orchestration* describing partial functionality required from other web services.

**Mediators** describe elements that aim to overcome structural, semantic or conceptual mismatches that appear between the different components that build up a WSMO description. WSMO specification currently covers four types of mediators: (1) *OOMediators* import the target ontology into the source ontology by resolving all the representation mismatches between the source and the target, (2) *GGMediators* connect goals that are in a relation of refinement and resolve mismatches between those, (3) *WGMediators* link Web services to goals and resolve mismatches, and (4) *WWMediators* connect several Web services for collaboration.

### 1.4.2 SWS Execution Environment (WSMX)

Based on WSMO concepts, the Web Services Execution Environment (WSMX) is the execution environment for discovery, composition, engagement, selection, mediation and invocation of Semantic Web Services. The global WSMX architecture and its components are depicted in figure 1.1.

**The WSMX Manager and the Execution Engine** facilitate a Semantic Web Services execution process (execution semantics) by triggering discovery, composition, engagement, selection, mediation and invocation components upon receiving users' request (goal) and within the whole interaction process between service requester and service providers. The WSMX execution semantics is the core of the WSMX intelligence providing value-added services to traditional communication. Different execution semantics can be used according to the domain-specific requirements, e.g. mediation components are only required for heterogeneous environments, a selection component is used when (semi) automated decisions to select the best services are required based on requesters' preferences, or specific execution semantics exists for registering ontology or service in WSMX repositories. Typically, execution semantics is triggered based on the WSMX system entry point invoked by external application (adapter, front-end application, etc.).

**The Resource Manager** is responsible for the management of repositories to store definitions of web services, goals, ontologies and mediators.

**Discovery and Composition** of web services is one of the key processes of the SWS technology. A number of services could be returned from this step, services which satisfy the goal composed to a predefined process (*once-for-all* composition) as well as dynamically created process (*on-the-fly* composition). Such composition of services could also include "duplicate" services with the same capabilities however with different characteristics (non-functional properties). For duplicate ones, selection of services will be performed.

**Engagement** is composed of two phases, namely *contracting* and *negotiation*. Usually, discovery and composition operate on more general (abstract) goals. They result with a set of web services that can potentially fulfill a requester goal. However, for a complete guarantee that discovered web services will be able to provide requested concrete service, communication between a requester and a provider is necessary. This phase is called contracting. In addition, negotiation with each perspective web service to reach agreement on terms of services can also be performed.

**Selection** of the best or optimal service is performed when a number of duplicate services is returned from the discovery and composition process. To find an optimal service, different techniques can be applied, ranging from simple selection criteria ("always the first") to more sophisticated techniques, such as multi-criteria selection of variants also involving interactions with a service requester. Different variants of services could be described by different values of parameters (non-functional properties specific to web services), such as financial properties, reliability, security, etc.

**Data and Process Mediation** facilitate interactions between two entities when different ontologies or different choreographies are used by these entities. Data Mediation is ensured by mappings between concepts from one ontology to another one. It is based on paradigms of ontology engineering, i.e. ontology mapping/aligning [**?**]. Process mediation provides the necessary functionality for a runtime analysis of two given choreography instances and compensate possible mismatches that may appear, for instance, grouping several messages into a single one, changing their order or even removing some of the messages.

**Invoker and Receiver** implement an entry point of the WSMX responsible for receiving of incoming requests and invoking web services respectively. Invoker and receiver also handle grounding of WSMO services to underlying WSDL and SOAP protocol.

**Adapters** lie outside of WSMX and facilitate the interoperability between a requester and a provider at the technical level. WSMX is designed to internally handle WSML messages encapsulated in WSDL and sent or received using SOAP protocol. Therefore, adapters must ensure that interactions between a service requester and provider can be performed using different communication protocols (e.g. FTP) as well as in different languages (such as XML).

# Chapter 2

# Use Case Scenario

As organizations rely more on collaboration with partners to enhance their competitiveness, integration of information systems is important in cutting costs in information exchange and enabling quicker reactions to e.g. demand fluctuations. B2B integration is not however easy as companies have heterogeneous information systems that cannot be integrated easily. There are certain standards for B2B integration such as EDI X.12 and RosettaNet that are meant to ease the integration task by providing guidelines how certain processes are integrated [PCB$^+$05]. RosettaNet is a newer standard, which is constantly growing in the significance. RosettaNet had over 3000 documented production implementations in 2004 with a big growth on a way [Dam04].

However, the B2B integration even when using an XML-based standards, such as RosettaNet, can currently take six months[PCB$^+$05]. This is due to the flexibility of these standards regarding message content, message sequencing and message security details which are often not formally defined. This means that just supporting a same message does not mean that companies become interoperable. A significant effort is required to agree and implement exactly how the standard is used. This leads to the fact that current B2B integration partnerships are still quite rigid and longstanding.

The scenario assumes that SWS technologies come to B2B integration stepwise instead of big bang. So first SWS environments are used with existing B2B protocols and the SWS based B2B integrations can only after e.g. popular B2B protocols available use SWS technologies in their specifications. If all companies would start using formal domain ontologies, it would eliminate the need for these complex adapters, but this seems to be still a big if as even the existing B2B frameworks are not defined using SWS languages. Our scenario was based on the idea that existing RosettaNet and EDI implementations are working and they will not be substituted easily. So instead of making scenario of partners using purely SWS in the communication, we assumed that the B2B integration still use existing working solutions from B2B protocols e.g. for security. This working with older standards for B2B integration means that this solution can more easily be verified in real business environments, but still the use of SWS can help in B2B integrations.

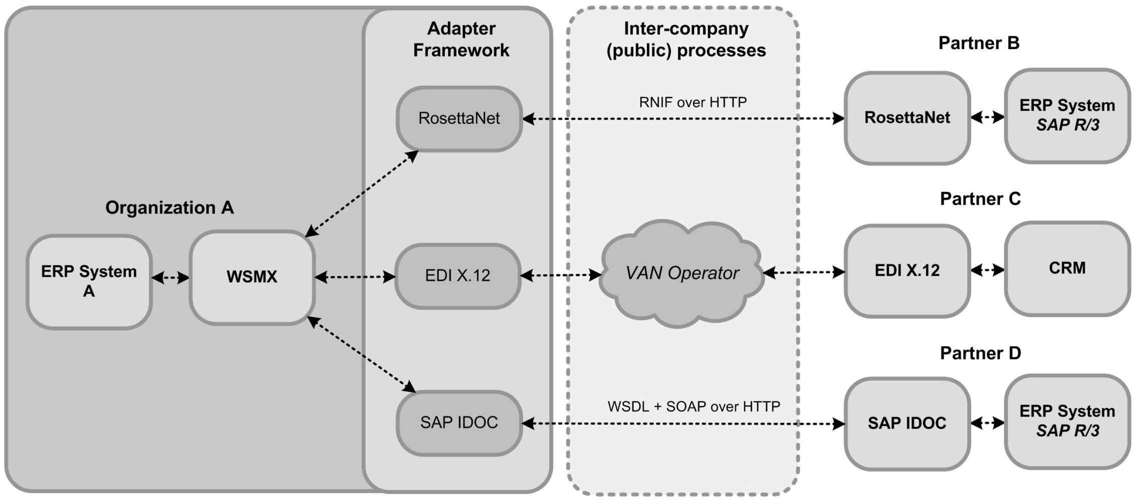

Figure 2.1: Use Case Scenario

## 2.1 Use Case Description

In our use case depicted in figure 2.1, we consider the buyer organization A which manufacturers electronic devices. For particular device, this organization needs specific component, in our case a *display unit X*. This display unit can be delivered by three different suppliers (further referred to as partners), namely B, C and D. The organization intends to build the B2B integration with all these partners and make preliminary agreement on possible trades. As the partners are large companies, the organization cannot just dictate the way how the B2B integration should happen. Therefore, the organization has implemented separate B2B integration with each partner, namely RosettaNet using RosettaNet Implementation Framework (RNIF) over HTTP with partner B, EDI X.12 using Value-Added Network operator over specific network communication with partner C and SAP Intermediate Documents (SAP IDOC) using Web Services standards with partner D.

The organization utilizes WSMX SWS technology for the B2B integration with all its partners. The ERP system of the organization is integrated with WSMX the way that requests in a form of a WSML goal are sent directly to WSMX by invoking specific WSMX system interface (in our case, *achieveGoal* system interface). Integration of the organization and each B2B protocol of each supplier should be built using a specific *adapter*. The role of each adapter is to adapt all protocols and languages used in specific communication to WSML language and SOAP protocol used by the WSMX technology (*technical-level interoperation*). Later in this deliverable we show in detail the functionality of RosettaNet adapter as part of the Semantic Web Services execution process. WSMX further enables data mediation of different messages used in integration of different B2B protocols (*data-level interoperation*) and process mediation of different communication patterns (*process-level interoperation*).

In order to enable successful interoperation of the organization and its partners, integration at all these three levels is needed. Following is the description of all these levels for our use case.

**Technical-level interoperation** is an ability of companies to exchange messages using specific communication protocols as well as languages. In our use case, following inconsistencies exist at technical level:

- partner B uses RosettaNet Implementation Framework (RNIF 2.0) over HTTP(S) for secure transport. RNIF guides how the messages are sent and acknowledged and how digital signatures are used.
- With partner C the transportation uses a VAN operator, who takes care of messaging between the partners. Basically the messages are dropped to a file system folder, where the VAN operator collects the messages and takes care of sending to the partner. As often is the case with EDI messages are not acknowledged but they are assumed to have gone through.
- With partner D, the transportation uses username/password secured Web Services over HTTP(S). If something goes wrong the web services responses with an error code.

**Data-level interoperation** is an ability of companies to understand exchanged messages. In our use case following inconsistencies exist at data (semantic) level.

- Partner B uses the RosettaNet Partner Interface Process (PIP) messages according to the message guidelines provided by RosettaNet. So the *PIPs 3A1 Request for Quote* and *3A4 Request Purchase order* are used. Both consist of exchanging request and response messages.
- Partner C supports EDI X.12 messages and expects the *840 Request for Quotation* for queries and *850 Purchase Order* for orders. It responds price and availability queries by *879 Price Information* message and to purchase orders by *855 Purchase Order Acknowledgment* message. These differ from semantics from the RosettaNet messages.
- Partner D uses SAP IDOCS for these. E.g. ORDERS05 IDoc for PO information exchange.

**Process-level interoperation** is an ability of companies to exchange message in the right timing and sequence order. In our use case, following inconsistencies exist at process level.

- Partner B complies again with RosettaNet choreographies in quoting and purchase order processing. That means the response message to queries arrives within 24 hours of sending of the requests. If the response message is not in time the PIP will automatically report an error.

- Partner C with EDI has not such fixed response times between different messages is not dictated by EDI. However, to serve the organization needs they answer these messages within 24 hours of the request although due to transportation the actual delivery can take an extra hour or so.
- The implementation of partner D has no set timers but they also are committed to 24 hours in response. Partner D does not have any automatic time-out either. The response is sent directly to the organization A external web service so there are no delays on the way.

## 2.2 Integration Process

The whole integration process of organization A and partners B, C, and D happens in two phases: (1) *integration set-up phase* and (2) *integration runtime phase*. During the set-up phase, the integration is built based on requirements and design, and in runtime phase, a semantic web service execution process happens. In a nutshell we envision this process for our scenario as (1) organization A sends its request from its ERP system (as WSML goal) to the WSMX requesting 10 pieces of display unit X devices to be purchased and delivered. In WSMX, (2) possible suppliers capable of fulfilling this request are discovered, (3) engagement with these suppliers is made to get price and conditions for a trade, (4) the best supplier is selected based on the price and conditions, (5) purchase order is submitted to this supplier and (6) result is sent back to the ERP system.

Further in this deliverable we describe in detail how integration set-up phase looks like for the RosettaNet as well as describe in detail SWS execution process.

# Chapter 3

# Semantic Web Services and B2B Integration Process

Semantic Web Services and B2B integration process is divided into two phases, namely *integration set-up phase* and *integration runtime phase*. In this chapter we describe these phases in detail with respect to the use case scenario described in the previous chapter.

## 3.1 Integration Set-up Phase

Integration set-up phase needs to be done before any integration between companies can happen. This phase is specific for particular B2B protocols used by each partner and requires prior agreement on formats of messages used in communication. According to this agreement, specific interoperation is implemented and remains proprietary for this particular integration. Following are the stages for integration set-up phase defined for the WSMO/WSML/WSMX SWS concepts and a B2B protocol. In addition, we demonstrate this phase on our case scenario and in particular on RosettaNet B2B protocol used between organization A and partner B.

### 3.1.1 Analysis of Requirements

During this stage, analysis of requirements should be performed for specific B2B protocols and conditions specified by both partners. Due to the flexibility of B2B protocols regarding message content, message sequencing and message security details which are often not formally defined, prior agreement on specific formats must be achieved between business partners. Companies may often have their own specification available for particular B2B protocol they are using.

Following outcomes should result from this stage.

- B2B protocol used (e.g. RosettaNet).
- A set of B2B transactions used including interaction processes and messages used (e.g. RosettaNet PIP 3A1 and related messages).
- Concrete agreement on format of messages.

In our case scenario, during the analysis of requirements we presume, RosettaNet *3A1 PIP* have been identified as well as general agreement on the precise format used has been made between organization A and partner B. We do not show here 3A1 DTD schema for this message due to its size, we only show 3A1 instance message as for example.

```
<?xml version="1.0" encoding="UTF−8"?>
<!DOCTYPE Pip3A1QuoteRequest SYSTEM "3A1_MS_V02_01_QuoteRequest.dtd">
<Pip3A1QuoteRequest>
  <fromRole>
    <PartnerRoleDescription>
      <GlobalPartnerRoleClassificationCode>Buyer</GlobalPartnerRoleClassificationCode>
      <PartnerDescription>
        <BusinessDescription>
          <GlobalBusinessIdentifier>111111111</GlobalBusinessIdentifier>
        </BusinessDescription>
        <GlobalPartnerClassificationCode>Manufacturer</GlobalPartnerClassificationCode>
      </PartnerDescription>
    </PartnerRoleDescription>
  </fromRole>
  <GlobalDocumentFunctionCode>Request</GlobalDocumentFunctionCode>
  <Quote>
    <GlobalQuoteTypeCode>Bid for Buy</GlobalQuoteTypeCode>
    <QuoteLineItem>
      <comments>
        <FreeFormText>Looking for best price for component GTIN 12345678901234) for the delivery date
            specified</FreeFormText>
      </comments>
      <GlobalProductUnitOfMeasureCode>Piece</GlobalProductUnitOfMeasureCode>
      <isSubstituteProductAcceptable>
        <AffirmationIndicator>No</AffirmationIndicator>
      </isSubstituteProductAcceptable>
      <LineNumber>1</LineNumber>
      <ProductIdentification>
        <GlobalProductIdentifier>12345678901234</GlobalProductIdentifier>
      </ ProductIdentification >
      <requestedQuantity>
        <QuoteQuantity>
          <QuantityTransportationSchedule>
            <GlobalTransportEventCode>Dock</GlobalTransportEventCode>
            <QuantitySchedule>
              <DateStamp>31122005</DateStamp>
              <ProductQuantity>10</ProductQuantity>
            </QuantitySchedule>
          </QuantityTransportationSchedule>
        </QuoteQuantity>
      </requestedQuantity>
      <shipTo>
        <PartnerLocationDescription>
          <BusinessDescription>
            <businessName>
              <FreeFormText>Galway plant</FreeFormText>
            </businessName>
          </BusinessDescription>
          <GlobalPartnerClassificationCode>Manufacturer</GlobalPartnerClassificationCode>
          <PhysicalLocation>
            <PhysicalAddress>
              <addressLine1><FreeFormText>IDA Business park</FreeFormText></addressLine1>
              <addressLine2><FreeFormText>Lower Dangan</FreeFormText></addressLine2>
              <cityName><FreeFormText>Galway</FreeFormText></cityName>
              <GlobalCountryCode>IE</GlobalCountryCode>
            </PhysicalAddress>
          </PhysicalLocation>
        </PartnerLocationDescription>
      </shipTo>
    </QuoteLineItem>
  </Quote>
  <thisDocumentGenerationDateTime>
    <DateTimeStamp>20051213−1013</DateTimeStamp>
  </thisDocumentGenerationDateTime>
  <thisDocumentIdentifier>
    <ProprietaryDocumentIdentifier>20051213−1013@organizationA</ProprietaryDocumentIdentifier>
  </thisDocumentIdentifier>
  <toRole>
    <PartnerRoleDescription>
      <GlobalPartnerRoleClassificationCode>Distributor</GlobalPartnerRoleClassificationCode>
    </PartnerRoleDescription>
  </toRole>
</Pip3A1QuoteRequest>
```

Listing 1: RosettaNet PIP 3A1 instance message

### 3.1.2 Design and Implementation

During this stage, *ontologizing, design and registration of mapping rules for data mediation* and *design and implementation of adapters* is performed. Following outcomes should result from this stage.

- Ontologies for identified messages and rules for runtime phase for transformation of messages to these ontologies.
- Adapters for identified B2B protocols.

#### Ontologizing

Based on requirements, *ontologies* for identified B2B protocols and messages should be created as well as *rules for runtime phase for transformation of selected B2B messages to these ontologies and vice versa* should be created. For these purposes, specific tools can be used/developed allowing for example semi-automated transformation from XML Schemes and RosettaNet messages to WSML ontology language. We call this transformation *ontologizing* and it requires an expert who understands specific B2B protocol as well as has good understanding of ontology languages to be able to capture information in messages ‘semantically’.

When ontologizing, *domain ontology* with respect to all used messages and all B2B protocols should be developed by the buyer organization for its own purposes taking into account all supported B2B protocols. Alternatively, such domain ontology could be reused and adopted by the organization if one already exists. Ideally, domain ontology is adopted by all business partners which intend to utilize semantic integration. All messages of B2B protocols are then ontologized with respect to the domain ontology.

In addition, newly created ontologies are registered with WSMX as well as mapping rules between newly created ontologies and selected existing ontologies in the WSMX ontology repository should be found. The necessity for these mappings is based on what is needed in current situation. For example, in case a domain ontology exist for all B2B protocols used by the organization, any mappings might not be needed as semantic interoperation has already been achieved by domain ontology. Registration of ontologies is done by calling registration entry point of the WSMX system interface using a specific client tool for this purpose. Finding of mappings for data mediation is facilitated by the data mediation tool. Both tools are part of the WSMX Management Tools called WSMT (Web Services Modeling Toolkit) [1].

In this deliverable, we are not focused on precise process of ontologizing and creating a domain ontology for B2B protocols, we will only show result of such transformation for RosettaNet PIP 3A1 message captured in WSML ontology as shown in listing 2. As mentioned earlier, design-time tools as well as detail methodology for ontologizing of RosettaNet messages for this purpose could be used if they exist. Creation of such tools or methodologies is out of the scope of the work of this deliverable, we only show the result of such process - PIP 3A1 message captured in WSML ontology language. The 3A1 ontology in WSML is provided in appendix 1.

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml−syntax/wsml−DL"
namespace { _"http://www.wsmo.org/RFQ#",
                     dc _"http :// purl.org/dc/elements/1.1#",
                 xsd _"http :// www.w3.org/2001/XMLSchema#" }
ontology RFQ

instance QuoteforProductX memberOf {RFQ}
nonFunctionalProperties
  dc#description hasValue "RFQ instance example."
endNonFunctionalProperties
  buyer hasValue "11111111"
        globaldocumentfunctioncode hasValue "Request"
        quote hasValue Quote
        thisdocumentgenerationdatetime hasValue "2005,12,13,10,13,23,0"
        thisdocumentidentifier  hasValue "20051213101323001@companyA"
        seller  hasValue "222222222"

instance Quote memberOf QuoteforProductX
        globalpartnerclassificationcode  hasValue "Manufacturer"
        lineitem  hasValue quoteLineItem
        QuoteTypeCode hasValue "Bid for Buy"

instance quoteLineItem memberOf QuoteforProductX
        comments hasValue "Looking for best price for display unit  device X (GTIN 12345678901234) for the
              delivery date specified"
```

[1]http://sourceforge.net/projects/wsmt

```
          isSubstituteProductAcceptable hasValue ”No”
          linenumber hasValue ”1”
           productidentification  hasValue ”12345678901234”
          requestedquantity hasValue requestedQuantity
          shipto  hasValue shipToLocation

instance requestedQuantity memberOf QuoteforProductX
          globalTransportEventCode hasValue ”Dock”
          dateStamp hasValue ”2005,12,31”
          productQuantity hasValue ”10”

instance shipToLocation memberOf QuoteforProductX
          businessdescription hasValue businessDescription
           globalpartnerclassificationcode  hasValue ”Manufacturer”

instance businessDescription memberOf QuoteforProductX
          businessname hasValue ”Galway plant”
           physicallocation  hasValue physicalLocation

instance physicalLocation memberOf QuoteforProductX
          physicaladdress hasValue physicalAddress

instance physicalAddress memberOf QuoteforProductX
          addressline hasValue ”IDA Business park, Lower Dangan”
          cityName hasValue ”Galway”
          countryCode hasValue ”IE”
```

Listing 2: PIP 3A1 message in WSML

### Adapters

The next step of the design stage is the design of adapters. Adapters are used when WSMX which operates on WSML needs to interact with other applications to achieve interoperation at the technical level. The use of system specific languages and data formats such as RosettaNet, IDOCS and EDI X.12 thus introduces an integration challenge. Concrete adapters must be developed for concrete B2B protocols in order to be integrated with WSMX. In nutshell, adapters should provide (1) *registration interface* for a partner, (2) *communication interface with a partner* according to the specific B2B protocol used, (3) *communication interface with WSMX* for integration of adapter with WSMX, (4) *B2B protocol transformations from/to WSML*, (5) *B2B protocol core functionality*.

Following is detail description of these components with respect to our scenario and RosettaNet.

- **Registration interface** allows a partner to register with the organization and provide necessary information in order to be able to communicate and exchange messages in RosettaNet. We define this registration interface as

  $$register(pip_1, pip_2, ..., pip_n, role, endpoint, certificate)$$

  where

  - $pip_1, pip_2, ..., pip_n$ is a set of PIPs used by the partner in the communication. All these PIPs are subject to the analysis and design between both, the partner and the organization.
  - $role$ is the specification of the role (*seller* or *buyer*) for the partner in the communication defined by the set of PIPs.
  - $endpoint$ is the *IP address* and *port* of endpoint interface of the partner where all messages will be sent from the adapter. All communication between the partner and the organization is done asynchronously.
  - $certificate$ is the public certificate of the partner to be used for digital signing of messages.

  By invoking this registration interface by the partner, following actions are performed in the adapter:

  - A service based on set of PIPs and role is created and described in WSML. PIPs are used for describing capabilities and interfaces of the service.
  - Ontologies for messages of these PIPs which are used by the service must be in place and registered with WSMX. These ontologies were previously created as a result from ontologizing.
  - The service is registered with WSMX by calling WSMX registration entry-point.

  In our example, partner B registers two PIPs, namely PIP 3A1 (Request for Quote) and PIP 3A4 (Purchase Order) and the *seller* role. In this case, PIP 3A1 is used as part of description for the choreography interface for engagement and PIP 3A4 is used as part of description of the choreography interface for invocation (see section 3.2 for more details).

- **Communication interface with WSMX** for sending and receiving WSML messages between WSMX and the adapter. This interface is the web service interface providing operations used for sending and receiving WSML messages from WSMX to the adapter and vice versa. This interface is then used for grounding of WSML concepts in choreography definitions.

- **Communication interface with a partner** for sending and receiving B2B specific messages. In our case, this communication interface is the endpoint interface provided by the RosettaNet adapter which allows the partner to send messages in RosettaNet format and using RNIF for communication.

- **B2B protocol transformation from/to WSML** for messages based on transformation rules of ontologizing.

- **B2B protocol core functionality** is the specific functionality needed to be provided by the adapter. In our case of RosettaNet adapter this is validation of RosettaNet messages, signing messages using public keys from certificates, creating RNIF envelopes, etc. The B2B functionality to be put to adapters is a matter of balancing with WSMX. RosettaNet adapter is in fact a wrapper for RosettaNet B2B protocol being used for communication with the partner. This wrapper provides a Semantic Web Service registered in WSMX. In general, the more RosettaNet functionality is managed to be generalized and transformed to the semantic level of WSMX, the more intelligence in WSMX can be applied. In this case, e.g. signing messages using public keys is done in this adapter however one can imagine that this would fit to the WSMX security and privacy. Security and privacy is currently not tackled in WSMX.

## 3.2 Integration Run-time Phase

After the integration set-up phase between the organization and its partners is finished, the environment is ready for the run-time phase. In this section we first describe the whole execution process as it happens in WSMX according to our scenario and then in detail we describe concreate functionality of the RosettaNet adapter and interactions within this process.

### 3.2.1 Execution Process

In our scenario, following stages happen as part of the WSMX execution process: (1) *Sending goal*, (2) *Discovery*, (3) *Engagement*, (4) *Selection*, (5) *Invocation*, (6) *Receiving results*. The activity diagram is for the run-time phase and our scenario depicted in figure 3.1.

- **Sending Goal**. Company A sends its request as a WSML goal from its back-end application (ERP system) to the WSMX entrypoint $achieveGoal$. The request is to *purchase and get delivered 10 pieces of display unit device X to Galway plant in Ireland*. In our scenario, we assume organization A is using SAP R/3 system, thus the integration of SAP and WSMX is built inside the SAP system. A specific user interface/integration with existing SAP functionality must be also in place in the SAP system so that users are allowed to specify such requests. Detail integration of back-end application with WSMX is out of the scope of this deliverable.

- **Discovery**. After the goal is received by WSMX, semantic web services execution process starts with invocation of the WSMX discovery component. All services matching this request which have been previously registered in WSMX are found (registration of services happened during the integration set-up phase). In

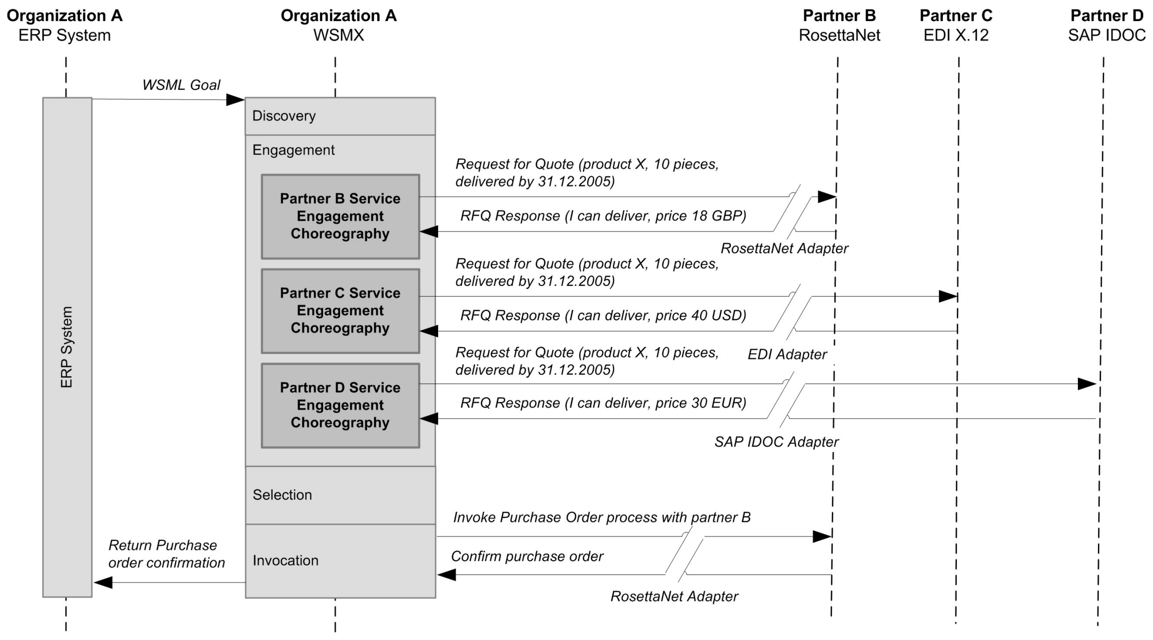
Organization A
ERP System
Organization A
WSMX
Partner B
RosettaNet
Partner C
EDI X.12
Partner D
SAP IDOC
WSML Goal
ERP System
Discovery
Engagement
Partner B Service Engagement Choreography
Partner C Service Engagement Choreography
Partner D Service Engagement Choreography
Request for Quote (product X, 10 pieces, delivered by 31.12.2005)
RFQ Response (I can deliver, price 18 GBP)
RosettaNet Adapter
Request for Quote (product X, 10 pieces, delivered by 31.12.2005)
RFQ Response (I can deliver, price 40 USD)
EDI Adapter
Request for Quote (product X, 10 pieces, delivered by 31.12.2005)
RFQ Response (I can deliver, price 30 EUR)
SAP IDOC Adapter
Selection
Invoke Purchase Order process with partner B
Confirm purchase order
Return Purchase order confirmation
Invocation
RosettaNet Adapter

Figure 3.1: Run-time phase Interactions

our scenario, services of partners B, C, and D are discovered (we also presume, that all these services cover purchasing as well as shipment of products to requested address). During discovery, data mediation can be performed in case that concepts/ontologies used in the WSML goal differ from concepts/ontologies used in service descriptions. We don't describe discovery of services in this deliverable, discovery is the subject of research in DIP, ASG and also in Knowledge Web in other tasks of WP2.4.

- **Engagement**. As discovery operates on abstract description of services, the next step is to find out whether each discovered service is capable of providing a concrete service and under which terms and conditions. For this purpose, each registered service must contain choreography interface for engagement which is automatically used. The purpose of engagement is to "ask" all possible partners whether they can deliver required product within the given time and for what price. Using implemented functionality of particular adapters used by each partner, this information is gathered from partners using B2B protocols they support.

  In our case scenario, engagement is performed for partners B, C and D by sending request for quote to all companies. On result, quote from all partners is received. In section 3.2.2 we describe detail interactions between WSMX and partner B using RosettaNet by means of RosettaNet adapter.

- **Selection** Based on the information provided from engagement, the best service is selected. In our case this is done automatically according to the cheapest service based on price information received in the quotes. In more elaborated scenarios, selection of the ”best” services could be done in semi-automatic fashion when ”proposal” is presented to a user who makes the final approval. In our scenario, the partner B is selected which means that a conversion of different currencies used in quotes must happen. We don’t discuss this feature in this deliverable, however such conversion could happen using conversion functions associated with mapping rules of different ontologies or transformation rules from ontologizing.

- **Invocation** During invocation, the actual service is invoked. In our scenario it means, that purchase order is sent to the partner B. The concrete interactions between WSMX and partner B is done in analogical way like in the case of engagement. This is in more detail described in the following section 3.2.2.

- **Receiving Results**. After the invocation of the service is finished, the result is sent back to the system/user.

### 3.2.2 RosettaNet Adapter

In this section we describe specific functionality of the RosettaNet adapter within the execution process as it happens in WSMX. In particular, we describe interactions for engagement when after the service is discovered, it must be determined whether discovered service is capable of providing a concrete/requested service and under which conditions (price, delivery date, etc.). Eventhough we describe these interactions for engagement, the analogical interactions happen during invocation as well.

Interactions defined by this choreography are depicted in figure 3.2 as part of the block ”partner B Service Engagement Choreography”. RosettaNet adapter represents the semantic web service registered with WSMX and all interactions between WSMX and this adapter is based on web service communication as defined by *communication interface with WSMX* in section 3.1.2. That means, that information from/to WSMX is sent out/received by invoking specified operations of RosettaNet adapter web service as specified by grounding of the choreography interface.

Following is the detail description of interactions between organization A and partner B according to the figure 3.1.

- In WSMX, all information related to the quote is sent from WSMX to the adapter. This information includes *product name* (display unit X), *amount* (10 pieces), *requested delivery date* (31.12.2005). This information is captured in WSML.

- RFQ is received by RosettaNet adapter and based on this information, RosettaNet message 3A1 RFQ is created and validated against 3A1 message schema.

- In RosettaNet adapter, RNIF envelope for this message is created.

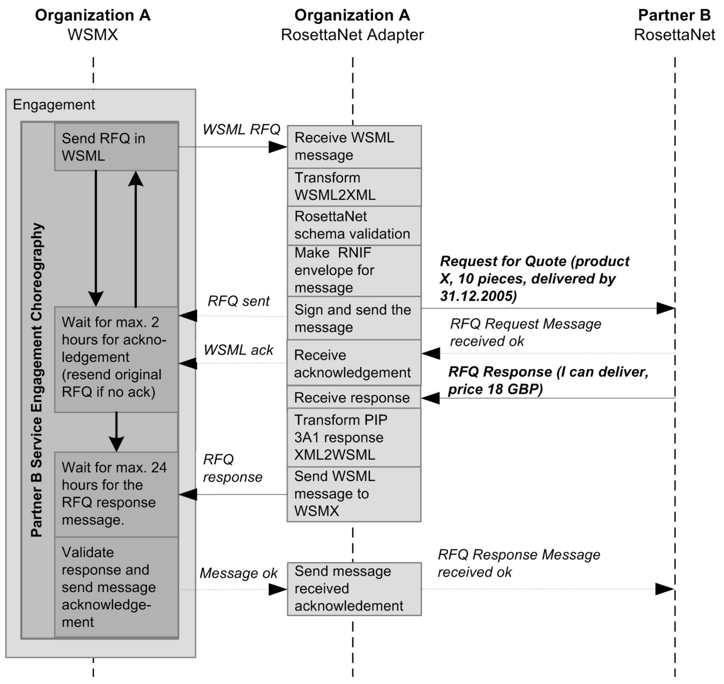
Organization A
WSMX
Organization A
RosettaNet Adapter
Partner B
RosettaNet
Engagement
Partner B Service Engagement Choreography
Send RFQ in WSML
Wait for max. 2 hours for ackno-ledgement (resend original RFQ if no ack)
Wait for max. 24 hours for the RFQ response message.
Validate response and send message acknowledge-ment
WSML RFQ
RFQ sent
WSML ack
RFQ response
Message ok
Receive WSML message
Transform WSML2XML
RosettaNet schema validation
Make RNIF envelope for message
Sign and send the message
Receive acknowledgement
Receive response
Transform PIP 3A1 response XML2WSML
Send WSML message to WSMX
Send message received acknowledement
Request for Quote (product X, 10 pieces, delivered by 31.12.2005)
RFQ Request Message received ok
RFQ Response (I can deliver, price 18 GBP)
RFQ Response Message received ok

Figure 3.2: RosettaNet Interactions

- In RosettaNet adapter, 3A1 message is signed using the public key from certificate of partner B and message is sent out to the endpoint of partner B (certificate as well as endpoint have been previously registered by partner B during the integration set-up phase). As a result, confirmation that mesage has been sent is sent back to the WSMX.
- In WSMX, after the confirmation is received, WSMX next expects acknowledgment of reception of message by partner B. Exceptional behavior is implemented by WSMX choreography, i.e. if partner B fails to acknowledge the message, it is sent again after 2 hours and up to 3 attempts are made. After that if acknowledgment is not received, service is cancelled.
- In WSMX, after the acknowledgment has been received, WSMX gets to the state where it waits for the RFQ response from partner B. Here, exceptional behavior is again implemented: if RFQ response is not received until specified time of 24 hours, processing of partner B service is cancelled assuming that partner B is not interested in providing the service.
- In RosettaNet adapter, RFQ response is received from partner B including information that devices can be delivered in specified time for the price of 18 GBP. This response is sent to the WSMX.
- In WSMX, response is received and acknowledgment is sent out through the adapter to partner B.

# Chapter 4

# Conclusions and Future Work

This is the first version of the deliverable describing invocation and interoperation of web services. Within the context of interoperation layers we discussed in particular the problem of the technical interoperation taking into account different B2B protocols used by different parties involved in inter-enterprise integration settings. These tasks are within the SWS execution environment covered mainly by adapters and techniques of ontologizing of particular B2B protocols. With this respect, we presented a scenario for B2B integration, where a buyer organization communicates with multiple partners, which all behave a bit differently. The partners support different B2B protocols for this communication. In the scenario, requests for quotes (RFQ) and purchase order (PO) processes are used. We showed how the interoperation aspects are handled at the buyer organization point of view, where the use of Semantic Web Service (SWS) technologies enables communication with all its business partners.

In the final version of the deliverable we plan to update the integration scenario by more selection mechanisms to address some industrial comments on the deliverable regarding that the selection was really simple. We will also add more details on the interactions from internal ERP systems point of view and provide detail description of how WSMX goals are obtained and include capabilities and choreography interfaces for engagement and invocation. We plan to enhance the current functionality of WSMX including implementation of engagement and invocation components as well as adapters for RosettaNet and other selected B2B protocols. This detail design proposals will provide the basis for the prototype implementation of this scenario with use of existing WSMX functionality which is being developed in other projects such as DIP [DIP06] and Lion [LIO06].

# Appendix 1: PIP 3A1 WSML Ontology

```
wsmlVariant _"http://www.wsmo.org/wsml/wsml−syntax/wsml−rule"
namespace { _"http://www.wsmo.org/RN#",
                          dc _" http :// purl.org/dc/elements/1.1#",
                     loc _" http :// www.wsmo.org/ontologies/location#",
                     xsd _" http :// www.w3.org/2001/XMLSchema#" }

ontology RFQ

axiom globaldocumentfunctioncodeConstraint
      nonFunctionalProperties
            dc#description hasValue "The type of globaldocumentfunctioncode can only be 'Request' or 'Response'"
      endNonFunctionalProperties
      definedBy
            !− ?x[globaldocumentfunctioncode hasValue ?type] memberOf RFQ
   and (?type = "Request"
   or
?type = "Response").

axiom globalpartnerroleclassificationcodeConstraint
      nonFunctionalProperties
            dc#description hasValue "The type of GlobalPartnerRoleClassificationCode can only be 'Buyer' or '
                   Seller' "
      endNonFunctionalProperties
      definedBy
            !− ?x[ globalpartnerroleclassificationcode  hasValue ?type] memberOf partnerInformation
   and (?type = "Buyer"
   or
?type = "Seller").

axiom globalsupplychaincodeConstraint
      nonFunctionalProperties
            dc#description hasValue
            "Code identifying the supply chain for the partner's function.
            − Electronic Components The electronic components supply chain.
            − Information Technology The information technology supply chain.
            − Semiconductor Manufacturing The semiconductor manufacturing supply chain.
            − Telecommunication IndustryLogistics Industry"
      endNonFunctionalProperties
      definedBy
            !− ?x[globalsupplychaincode hasValue ?type] memberOf partnerDescription
   and (?type = "Electronic Components" or ?type = "Information Technology" or
         ?type = "Semiconductor Manufacturing" or ?type = "Telecommunication Industry" or
         ?type = "Logistics Industry").

axiom globalPartnerClassificationCodeConstraint
      nonFunctionalProperties
            dc#description hasValue "Code identifying a partners function in the supply chain. Only parts of these
                   are now used for brevity. See RosettaNet for full descriptions
Broker, Carrier, Contract Manufacturer, Customs Broker, Distribution Center, Distributor , End User, End User
       Government,
Financier, Freight Forwarder, Manufacturer, Manufacturing Division, Original Equipment Manufacturer,
Reseller, Retailer, Service Provider,Shopper, Supplier, Warehouser."
      endNonFunctionalProperties
      definedBy
            !− ?x[globalpartnerclassificationcode hasValue ?type] memberOf globalpartnerclassificationcode
   and (?type = "Manufacturer"
   or
?type = "Supplier"
   or
?type = "Contract Manufacturer"
   or
?type = "Warehouser"
   or
?type = "Logistics Industry").


axiom businessDescriptionConstraint
      nonFunctionalProperties
            dc#description hasValue "At least one form of identification must be provided.
            This needs to be made to handle exceptions better in the future."
      endNonFunctionalProperties
      definedBy
            !− ?x[businessDescription hasValue ?type]
   and (?type = "businessname"
   or
?type = " globalbusinessidentifier "
   or
?type = " partnerbusinessidentification ").

axiom QuoteTypeCodeConstraint
      nonFunctionalProperties
            dc#description hasValue "Code identifying category specification for a quote.
Bid for Buy Customer ready to purchase, looking for best price.
Blanket Quote Secures pricing for complete parts lists
BOM Quote Complete list of parts provided for a specified project, total package pricing provided.
See others from RosettaNet"
      endNonFunctionalProperties
      definedBy
            !− ?x[QuoteTypeCodeConstraint hasValue ?type]
   and (?type = "Bid for Buy"
   or
?type = "Blanket Quote"
   or
?type = "BOM Quote").

axiom isSubstituteProductAcceptableConstraint
      nonFunctionalProperties
            dc#description hasValue "Indicates whether the requesting party will accept a substitute product for
                the product requested."
      endNonFunctionalProperties
      definedBy
            !− ?x[isSubstituteProductAcceptable hasValue ?type]
   and (?type = "Yes"
   or
?type = "No").

axiom globalTransportEventCodeConstraint
      nonFunctionalProperties
            dc#description hasValue "Dock The event representing when the product should arrive on the recipient's
                dock.
Pickup The event representing when the product is scheduled to be picked up from the shipper's dock.
Ship The event representing when the product is requested to ship."
      endNonFunctionalProperties
      definedBy
            !− ?x[globalTransportEventCode hasValue ?type]
```

```
  and (?type = ”Dock”
  or
?type = ”Pickup”
  or
?type = ”Ship”).

axiom physicalLocationConstraint
    nonFunctionalProperties
        dc#description hasValue ”Dock The event representing when the product should arrive on the recipient’s
             dock.
Pickup The event representing when the product is scheduled to be picked up from the shipper’s dock.
Ship The event representing when the product is requested to ship. ”
    endNonFunctionalProperties
    definedBy
        !− ?x[physicalLocation hasValue ?type]
  and (?type = ”Global Location Identifier ”
  or
?type = ”Physical Address”).

axiom countryCodeConstraint
    nonFunctionalProperties
        dc#description hasValue ” ISO 3166 Country Code. 2 letter versions. Not the complete list here.”
    endNonFunctionalProperties
    definedBy
        !− ?x[countryCodeCode hasValue ?type]
  and ( ?type = ”AT”
  or ?type = ”AU”
  or ?type = ”BE”
  or ?type = ”CA”
  or ?type = ”CH”
  or ?type = ”CZ”
  or ?type = ”DE”
  or ?type = ”DK”
  or ?type = ”ES”
  or ?type = ”FI”
  or ?type = ”FR”
  or ?type = ”GB”
  or ?type = ”HU”
  or ?type = ”IE”
  or ?type = ”IT”
  or ?type = ”KR”
  or ?type = ”LV”
  or ?type = ”NL”
  or ?type = ”PL”
  or ?type = ”PO”
  or ?type = ”PT”
  or ?type = ”RO”
  or ?type = ”RU”
  or ?type = ”SK”
  or ?type = ”US”
).


concept RFQ
nonFunctionalProperties
     dc#description hasValue ”Request for Quote Document (RosettaNet).
     Based on guidelines of PIP 3A1 version 3A1_MG_V02_01_00_QuoteRequest ( 22−May−2003).”
     dc#creator hasValue ”Paavo Kotinurmi”
endNonFunctionalProperties
          buyer ofType (1 1) partnerInformation
          globaldocumentfunctioncode ofType (1 1) _string
          quote ofType (1 1) Quote
          thisdocumentgenerationdatetime ofType (1 1)_dateTime
          thisdocumentidentifier ofType (1 1) _string
          seller ofType (1 1) partnerRole
```

```
concept partnerInformation
nonFunctionalProperties
        dc#description hasValue ”Basic information about the sender and receiver.”
endNonFunctionalProperties
        contactinformation ofType (0 1) contactInformation
        globalpartnerroleclassificationcode ofType (1 1) _string
        partnerdescription ofType (1 1) partnerDescription

concept partnerDescription
nonFunctionalProperties
dc#description hasValue ”The collection of business properties that describe a business identity . ”
endNonFunctionalProperties
        globalbusinessidentifier ofType (1 1) _int
        globalsupplychaincode ofType (0 1) _string
        globalpartnerclassificationcode ofType (1 1) globalPartnerClassificationCode

concept globalPartnerClassificationCode
        globalpartnerclassificationcode ofType (1 1) _string

concept partnerBusinessIdentification
nonFunctionalProperties
dc#description hasValue ”The collection of business properties that allow for the proprietary identification of
      a business entity .
ProprietaryBusinessIdentifier A unique business identifier assigned and administered by a private authority .
ProprietaryDomainIdentifier A descriptor that is used to categorize an organization or business entity that is
      in the Proprietary Business Identifier .
ProprietaryIdentifierAuthority A unique name that identifies an organization or business entity that is
      responsible for managing one or more lists of identifiers . ”
endNonFunctionalProperties
        proprietarybusinessidentifier ofType (1 1) _string
        proprietarydomainidentifier ofType (1 1) _string
        proprietaryidentifierauthority ofType (0 1) _string

concept Quote
nonFunctionalProperties
dc#description hasValue ”The collection of business properties that describe an offer to supply a
quantity of products at an agreed price and schedule.”
endNonFunctionalProperties
        comments ofType (0 1) _string
        globalpartnerclassificationcode ofType globalPartnerClassificationCode
        lineitem ofType quoteLineItem
        QuoteTypeCode ofType (1 1) _string


concept quoteLineItem
nonFunctionalProperties
dc#description hasValue ”The collection of business properties that describe a product entry in a
quote business document.”
endNonFunctionalProperties
        comments ofType (0 1) _string
        isSubstituteProductAcceptable ofType (0 1) _string
        linenumber ofType (0 1) _integer
        productidentification ofType productIdentification
        requestedquantity ofType (0 1) requestedQuantity
        shipto ofType (0 1) shipToLocation

concept productIdentification
nonFunctionalProperties
dc#description hasValue ”The collection of business properties that describe proprietary and global
 identifier information regarding a product.
One instance of either GlobalProductIdentifier or PartnerProductIdentification is mandatory.
Global Trade Identification Number (GTIN) is a global unique product identifier adopted by RosettaNet.”
endNonFunctionalProperties
        GTINidentifier ofType (0 1) _integer
```

```
            proprietaryIdentification  ofType (0 1)  proprietaryIdentification

concept proprietaryIdentification
nonFunctionalProperties
dc#description hasValue "An internal identifier used to identify a products."
endNonFunctionalProperties
     proprietaryProductIdentifier  ofType (1 1) _string
     productCodeOwner ofType (0 1) _string

concept requestedQuantity
nonFunctionalProperties
dc#description hasValue "The collection of business properties that describe the quantities associated with a
     product quote including schedules."
endNonFunctionalProperties
        globalTransportEventCode ofType (0 1) _string
        dateStamp ofType (0 1) _date
        productQuantity ofType (1 1) _integer

concept shipToLocation
nonFunctionalProperties
dc#description hasValue "The partner and/or location to which the product must be delivered."
endNonFunctionalProperties
        businessdescription ofType (1 1) businessDescription
        globalpartnerclassificationcode  ofType (1 1) globalPartnerClassificationCode

concept businessDescription
nonFunctionalProperties
dc#description hasValue "The collection of business properties that describe a business identity and location.
businessName The name of a business entity
 globalbusinessidentifier :  A unique business identifie , DUNS number of 9 digits or DUNS+4 indicating also the
     location.
partnerBusinessIdentification  The collection of business properties that allow for the proprietary
      identification  of a business entity .  "
endNonFunctionalProperties
        businessname ofType (0 1) _string
        globalbusinessidentifier  ofType (0 1) _integer
        partnerbusinessidentification  ofType (0 1) partnerBusinessIdentification
        physicallocation  ofType (0 1) physicalLocation

concept physicalLocation
nonFunctionalProperties
dc#description hasValue "Visiting address information, a gloval DUNS=4 identifier or address information. "
endNonFunctionalProperties
        GlobalLocationIdentifier  ofType (0 1) _string
        physicaladdress ofType (0 1) physicalAddress

concept physicalAddress
nonFunctionalProperties
dc#description hasValue "The visiting address. "
endNonFunctionalProperties
        addressline ofType _string
        cityName ofType (0 1) _string
        countryCode ofType (0 1) _string
```

Listing 1: PIP 3A1 WSML ontology concerning the PIP elements used in the scenario (the whole PIP 3A1 ontology would be still a lot bigger.

# Appendix 2: Standards for B2B integration

The standards in the scenario are here described in more detail. Also few other possible standards for B2B integration are briefly discussed.

## RosettaNet standard

RosettaNet is an industry-driven consortium aiming at creating, implementing, and promoting open e-business process standards [1]. The member organizations represent Information Technology, Electronic Components, Semiconductor Manufacturing, Telecommunications, and Logistics industries. The most important components standardized in RosettaNet are Partner Interface Processes (PIPs), dictionaries and RosettaNet Implementation Framework (RNIF).

Business process aspects. Partner Interface Processes (PIPs) specify standard process choreographies (public processes) for collaborating partners. Collaborating partners' internal processes interact with PIPs to initiate or receive business documents. So partners' intern processes can be as they like e.g. in own processes in different locations, but the external interface is standard according to PIP. Each PIPs define common inter-company public processes such as *PIP 3A1 Request Quote* and the associated business documents *Quote request* and *Quote confirmation*. Established PIPs, such as 3A1, contain a specification document, Document Type Definitions (DTD) and Message Guidelines (MG). A specification document defines the process with Unified Modeling Language (UML) activity and sequence diagrams and textual descriptions, the roles of the partners, time constraints for responses and necessary conditions to initiate messaging. RosettaNet processes are predefined and not executable [vdAK03]. The most recent PIPs use ebXML BPSS for specifying process information but those PIPs are still clear minority.

Business document aspects. Each PIP defines one or more business documents. The DTD and Message Guidelines (MG) define the PIP service content of one business document. The DTD defines the valid XML document structure of a PIP service content.

[1]http://www.rosettanet.org/

The MG introduces additional constraints and guidelines, such as what a modification date means and how the date value should be represented. RosettaNet business dictionary (RNBD) defines common terms used in all the PIPs and represents RosettaNet ontology. In addition to dictionaries, RosettaNet uses certain identifiers, such as Data Universal Numbering System (DUNS) codes to identify companies uniquely. PIP business documents can be considered as defining a common ontology, which is used while collaborating. Both collaborators need to translate their internal organizations ontology to this RosettaNet specified ontology, then there should be no misunderstandings. However, as there are voluntary issues, the interoperability can still be an issue in implementations. The most recent PIPs use W3C XML Schemas for specifying process information but those PIPs are still a clear minority.

Messaging aspects. RosettaNet Implementation Framework (RNIF) specifies messaging. It defines the RosettaNet business message that contains the service content specified by PIP DTD and MG, and the necessary headers and security features needed to process the messages. RNIF also defines how attachments are encoded in the RosettaNet business messages and uses e.g. MIME and S/MIME for packing and encryption. These attachments can be of arbitrary file format, such as MS/Word or AutoCAD. RNIF contains exception-handling mechanisms and makes sure that the delivery is non-repudiated, so neither the sender nor the receiver can later deny having sent/received the RosettaNet business message. RNIF features include e.g. that it does not handle messages further if it does not know the sender - this implies that initiating messaging cannot be totally dynamic. RNIF is peer-to-peer protocol between two collaborating partners. Many vendors, such as IBM, BEA and Microsoft, support RNIF in their products. RNIF does not use SOAP or other Web Service technologies.

RosettaNet does not guide contracting but in every PIP it assumes that collaborators have some kind of trading partner agreement in place. RosettaNet does not have any standards for registries or discoveries. In the RosettaNet web site [RosettaNet 2005] there are list of trading partners supporting certain PIPs and text-based searches.

## EDI

The development of Electronic Data Interchange (EDI) standards for B2B integration began already in the 1970s. There are two main EDI syntax standards in use. American National Standards Institutes (ANSI) ASC (Accredited Standards Committee) X12 EDI syntax is mainly used in North America. The EDI for Administration, Commerce and Transportation (EDIFACT) standard development is United Nations (UN) lead. EDIFACT is the dominant standard in the rest of the world. EDI is still the most used standard for B2B integration. See [FB05] for more on EDI and EDI ontology.

## SAP IDocs

SAP intermediate documents (IDocs) is central element in SAP. IDocs are meant to be used in integration of SAP to industrial standards, particularly EDI. Technically IDocs are very close to EDI messages, having segment information and data sections as in EDI. It borrows some EDI features and other transaction file formats. The document in IDoc refers to set of data comprising a functional group of records with a business identity. For example all the data in a PO is in ORDERS IDocs.

IDocs are used for asynchronous transactions: each IDoc generated exists as a self-contained text file, message, that can then be transmitted to the requesting endpoint. Since an IDoc is a message, both the sending and receiving applications must conform to a common convention about where, in a given IDoc, each piece of data will be found. To this end, SAP has defined several hundred IDoc types. And SAP owners can create their own custom IDoc. A sending application must construct an IDoc of a given type in accordance with these definitions. In the scenario used here, the IDoc semantics are mapped to semantics used in WSMX to represent RFQ and PO process information.

## Other standards for B2B integration

There are lots of other standards for B2B integration such as electronic business XML (ebXML), Open Applications Group Integration Specification (OAGIS) and Universal Business Language (UBL) that could be involved in the scenario. They have less reported actual implementations, but they are also valid standards that could be used in these kinds of scenarios. Therefor, they are also very briefly introduced. From here [NK04], you can read more on standards for B2B integration.

The mission of ebXML is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties [2]. EbXML is targeted for every sector of the business community, from international conglomerates to small and medium sized enterprises (SME). EbXML has defined a set of specifications designed to meet the common business requirements and conditions for e-business. EbXML has defined a set of modular specifications designed to meet the common business requirements and conditions for e-business. The ebXML Business Process Specification Schema (BPSS) is an XML-based specification language that can be used to formally define the public business processes that allow business partners to collaborate. ebXML Core Components specification defines common building blocks from which actual business documents can be built from. The ebXML messaging services (ebMS) provide a general-purpose messaging mechanism to allow reliability, persistence and security. It is very similar to RNIF in RosettaNet, but in many places is less strict. Collaboration Protocol Profiles and Agreement (CPPA) specification

[2]http://www.ebxml.org/

enables a company to specify its e-business capabilities. Agreement can be reached automatically Collaboration Protocol Profiles (CPPA) match. The ebXML Registry provides a set of services that enable sharing of information between interested parties. The two specifications describing the use of registries are Registry Information Model (RIM) and Registry Service Specifications (RSS). The information stored in registries are business process, documents, CPPA information or similar information. There are projects that use parts of the ebXML standard but the real ”‘in production”’ use implementations are still few.

The Open Applications Group [3] is an industrial consortium to create common standards for the integration of enterprise business applications. The OAG Integration Specification (OAGIS) defines common business documents and support for associated business processes. OAGIS defines Business Object Documents (BOD) as common message architecture. BODs are the business documents that are exchanged between collaborating partners information systems. BOD uses metadata to describe what kind of messages to expect. It is able to communicate status and error conditions in messaging. The OAGIS does not define many process aspects or secure messaging. BODs can be transported via protocols like HTTP or SOAP. OAGIS has published also reports on how OAGIS BODs can be trans-ported using RNIF 2.0. OAGIS has also promised support and provided documentation on using OAGIS BODs with ebXML BPSS and CPPA specifications. The OAGIS release 9.0 BODs are done according to ebXML core components guidelines.

Universal Business Language (UBL) [4]. The purpose of UBL is to develop a standard library of XML business documents (purchase orders, invoices, etc.) by modifying an already existing library of XML schemas, the xCBL. UBL 1.0 was declared an OASIS Standard in November 2004. The UBL 1.0 defines eight different business document schemas that belong to order-to-invoice procurement process. The UBL does not define many process aspects or secure messaging. UBL is concentrated in providing the standard business document schemas using ebXML Core Components guidelines.

[3]http://www.openapplications.org/

[4]http://www.oasis-open.org/committees/ubl/

# Bibliography

[CCMW01] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web service description language (wsdl) 1.1. Note, W3C, March 2001.

[Dam04] Suresh Damodaran. B2b integration over the internet with xml: Rosettanet successes and challenges. In *WWW Alt. '04: Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 188–195, New York, NY, USA, 2004. ACM Press.

[dBL$^+$05] Jos de Bruijn, Holger Lausen, et al. The Web Service Modeling Language WSML. Available from http://www.wsmo.org/2004/d16/. WSMO Working Draft v021, 2005.

[DIP06] Dip (data, information, and process integration with semantic web services), fp6 - 507483, 2006.

[FB02] Dieter Fensel and Christoph Bussler. The web service modeling framework wsmf. *Electronic Commerce Research and Applications*, 1(2):113–137, 2002.

[FB05] Doug Foxvog and Chris Bussler. Ontologizing edi: First steps and experiences. In *Proceedings of the International Workshop on Data Engineering Issues in E-Commerce*, 2005.

[GHM$^+$03] Martin Gudgin, Marc Hadley, Noah Mendelsohn, Jean-Jacques Moreau, and Henrik Frystyk Nielsen. SOAP version 1.2 part 1: Messaging framework. Recommendation, W3C, June 2003.

[LIO06] Lion project (http://lion.deri.ie/), 2006.

[Mar04] D. Martin. OWL-S: Semantic Markup for Web Services. Technical report, 2004. http://www.daml.org/services/owl-s/1.0/owl-s.html.

[NK04] Juha-Miikka Nurmilaakso and Paavo Kotinurmi. A review of xml-based supply-chain integration. *Production Planning and Control*, 15(6):608–621, 2004.

[PCB$^+$05] Chris Preist, Javier Esplugas Cuadrado, Steve Battle, Stuart Williams, and Stephan Grimm. Automated business-to-business integration of a logistics supply chain using semantic web services technology. In *ISWC '05: Proceedings of 4th International Semantic Web Conference*, 2005.

[POSV04] A. Patil, S. Oundhakar, A. Sheth, and K. Verma. Semantic Web Services: Meteor-S Web Service Annotation Framework. In *13th International Conference on World Wide Web,* pages 553–562, 2004.

[RLK04] D. Roman, H. Lausen, and U. Keller. Web Service Modeling Ontology (WSMO). WSMO Working Draft, 2004.

[vdAK03] W. M. P. van der Aalst and Akhil Kumar. Xml-based schema definition for support of interorganizational workflow. *Information Systems Research*, 14(1):23–46, 2003.

[WSD05] Web service semantics - wsdl-s w3c member submission, 2005.

[ZMH05] Michal Zaremba, Matthew Moran, and Thomas Haselwanter. Wsmx architecture. available from http://www.wsmo.org/tr/d13/d13.4/v0.2/. WSMX Final Draft, 2005.